Введение

Что нашёл:
https://infostart.ru/1c/articles/125344/
https://habr.com/ru/articles/271731
https://forum.mista.ru/topic.php?id=841622

Долго не искал. Статьи очень **давние**. Но и предлагаемый в этой статье поход к эмуляции ООП можно реализовать и на 1С8.0.

Интересны не столько статьи, сколько комментарии к ним. Часто – не нужен ОПП в 1С.
Из всех комментариев выбрал два

- ООП в 1С = резиновая баба
- А как же это ООП применить на практике

Эта статья написана в стиле «не лить много воды». Лаконично. Есть комментарии в коде.
И так. ООП - это *иерархия* классов.
Значит нам нужна какая-то иерархическая …
Можно «мутить» с XML, но нужно писать интерфейс.
Можно «мутить» с расширениями, но очень громоздко, хотя для типовых объектов можно обратить внимание.
Выбираем иерархический справочник. И так, класс — это элемент в иерархическом справочнике, а объект - структура. Вот и вся модель.

Принципы организации ООП будем рассматривать так, как это описано ниже.

Абстрагирование выделяет существенные характеристики некоторого объекта, отличающие его от всех других видов объектов и, таким образом, четко определяет его концептуальные границы с точки зрения наблюдателя.
Источник - https://ru.wikipedia.org/wiki/Объектно-ориентированное_программирование

Наследование - это принцип, который позволяет создавать иерархии классов: дочерние классы наследуют свойства и методы от родительских классов.

Инкапсуляция — это способ организации данных так, чтобы к ним нельзя было просто так обратиться извне.
Другими словами, инкапсуляция представляет собой способность объекта сохранять свои данные в приватной области видимости,
предоставляя для работы с ними только определенные публичные методы.
Таким образом, объект контролирует доступ к своим данным и обеспечивает их целостность.
Источник - https://skyeng.ru/magazine/wiki/it-industriya/chto-takoe-inkapsuliatciia/

Полиморфизм

- Полиморфизм подтипов (наследования)
- Параметрический полиморфизм (обобщённое программирование)
- Полиморфизм в интерфейсах

Источник - https://tproger.ru/articles/osnovnye-principy-oop-polimorfizm-v-programmirovanii
Все варианты полиморфизма реализуются в предлагаемой модели ООП.

Модель

Создаем справочник **Классы**.
Классы представляют собой элементы справочника **Классы.** Каждый элемент справочника (класс) имеет две ТЧ:

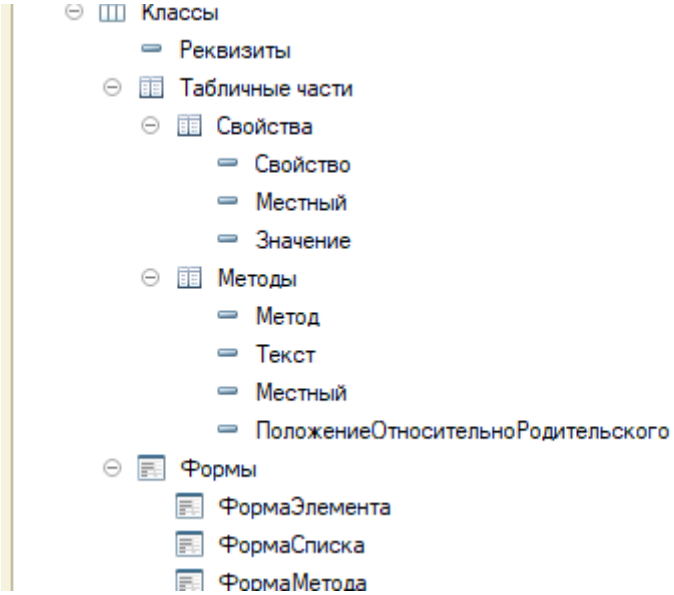
- Свойства
- Методы

Каждый класс имеет свой набор свойств и методов – строки в ТЧ.
Алгоритм создания объекта на основе класса следующий.

Объект — это структура. Объект создается в памяти.
Объект создается на основе конкретного класса (элемента справочника Классы).
При создании объекта добавляются все свойства и методы предков, которые не отмечены флагом Местный.
Если некоторый класс имеет метод и для этого метода реквизт Местный = Истина, то он не подлежит наследию.
Если класс имеет метод совпадающий по имени с наследуемыми методом, то он не переопределяется (остается таким, каким назначен для класса).
На языке расширений платформы 1С8 это «Вместо».
Это простеший случай. На самом деле уместно использовать нечто похожее на терминологию расширений.

```
Для каждого Стр Из Предок.Методы Цикл
    Если НЕ Стр.Местный ИЛИ Предок.Наименование = ИмяКласса Тогда //Инкапсуляция!!!
        //Местный метод не пойдет детям
        Если НЕ ЗначениеЗаполнено(Стр.ПоложениеОтносительноРодительского) Тогда
            СтруктураМетодов.Вставить(Стр.Метод,Стр.Текст);
        Иначе
            Если Стр.ПоложениеОтносительноРодительского = "До" Тогда
                КодМетода = Стр.Текст + Символы.ПС + СтруктураМетодов[Стр.Метод];
            ИначеЕсли Стр.ПоложениеОтносительноРодительского = "После" Тогда
                КодМетода = СтруктураМетодов[Стр.Метод] + Символы.ПС  + Стр.Текст;
            ИначеЕсли Стр.ПоложениеОтносительноРодительского = "Вместо" Тогда
                //Накрываем родительские
                КодМетода = Стр.Текст;
            ИначеЕсли Лев(Стр.ПоложениеОтносительноРодительского,5) = "Между" Тогда
                ПозицияВставки = СтрНайти(КодМетода,"//" + Стр.ПоложениеОтносительноРодительского);
                Если ПозицияВставки = 0 Тогда
                    Сообщить("Вставка не найдена");
                Иначе
                    КодМетодаДоВставки = Лев(КодМетода,ПозицияВставки-1);
                    КодМетодаПослеВставки = Сред(КодМетода,ПозицияВставки + 8);
                    КодМетода = КодМетодаДоВставки + Стр.Текст + КодМетодаПослеВставки;
                КонецЕсли;
            КонецЕсли;
            СтруктураМетодов.Вставить(Стр.Метод,КодМетода);
        КонецЕсли;
    Иначе
        // Предку достался метод, но он его детям не дает
        СтруктураМетодов.Удалить(Стр.Метод);
    КонецЕсли;
КонецЦикла;
```

Здесь интересно, что в расширениях один раз – «расширяемся».
А здесь конечный метод создается с помощью синтаксиса До, После, Между от предками к детям.
В итоге получается такой сборный метод.
Справочник классы в конфигураторе.



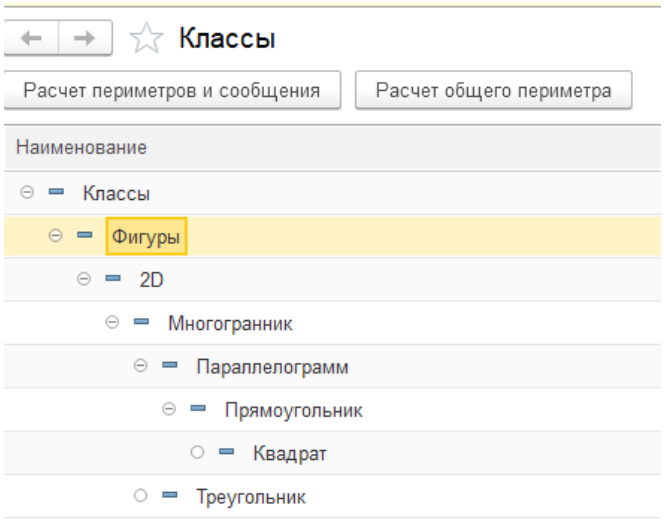
Здесь уместно уже оговорить, что конфигурация, прилагаемая к статье – написана для попытки наглядного изложения. Написана сугубо только для статьи. Без всяких проверок и наворотов. Есть продвинутый вариант «аля» ООП в 1С. Где не просто свойства объекта, а элементы XML. И свойства являются атрибутами элемента XML, текст элемента - метод. То есть Объект имеет методы как и в этой статье, но еще и методы элементов XML. Еще объект наделен массивами. Так выглядит модуль менеджера справочника **Классы**.

```
#Область ГеттерыСеттеры
//Значение
Функция Получить(ПолноеИмя)  Экспорт ...

Функция Установить(ПолноеИмя,Значение) Экспорт ...

//Массив
Функция УстановитьМассивы(ПолноеИмя,ИмяМассива="",Массивы) Экспорт ...

Функция ПолучитьМассив(ПолноеИмя,ИмяМассива="") Экспорт ...

//Элементы
Функция ПолучитьЭлемент(ПолноеИмя) Экспорт ...

//Методы
Функция ПолучитьМетодКласса(ИмяКласса,Метод) Экспорт ...

Функция ПолучитьМетодЭлемента(ИмяЭлемента) Экспорт ...

Функция ПолучитьМетод(ИмяКласса) Экспорт ...

Функция УстановитьМетодКласса(ИмяКласса,Метод,КодМетода) Экспорт ...

Функция ВыполнитьМетод(ОбъектКласса,Метод,Об="")  Экспорт ...

#КонецОбласти

#Область Имена

Функция ИмяАтрибута(ПолноеИмя) ...

Функция ИмяЭлемента(ПолноеИмя) ...

Функция ИмяОбъекта(ПолноеИмя) ...

Функция ПолноеИмяЭлемента(ПолноеИмя) ...

Функция ИменаДетейКласса (ИмяКласса) Экспорт ...

Функция ИменаМассивовКласса () Экспорт ...

Функция КоличествоМассивов() Экспорт ...

Функция АтрибутыКласса() Экспорт ...

#КонецОбласти

#Область Создание

Функция СоздатьОбъект(ИмяКласса)  Экспорт ...

Функция СоздатьДокумент(ИмяДокумента) Экспорт ...

Функция ПоискМассиваПредков(Родитель,МассивПредков) ...

#КонецОбласти
```

Здесь еще много планов по развитию в сторону «скрещивания» классического ОПП и XML. Также добавления перехватчиков событий для сеттеров и геттеров и так далее.

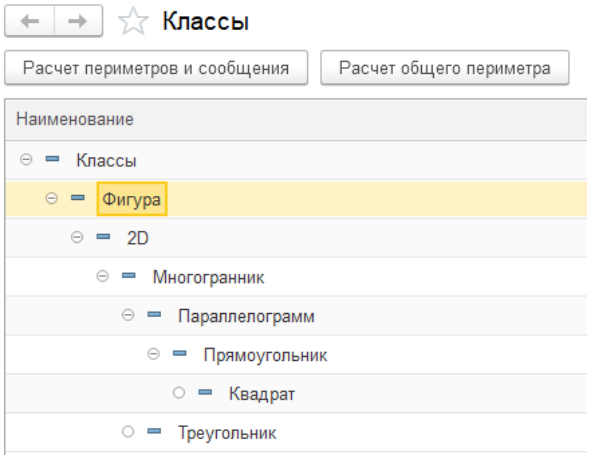
Это уже следущая статья… Такое развитие подход получил в результате примениения его на прпактике.

И так рассмотрим упрощенный вариант.

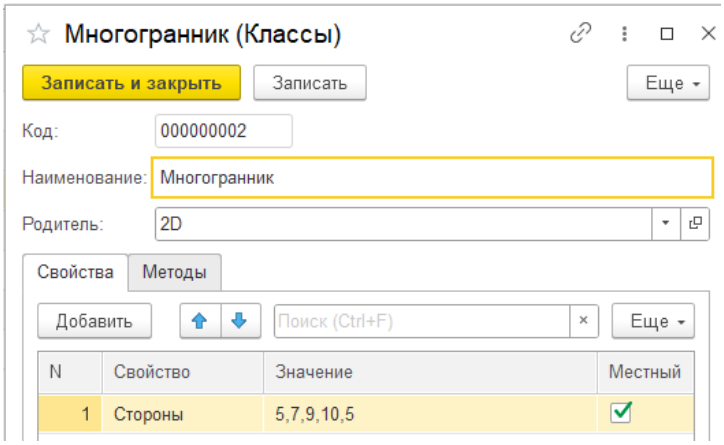


Классический пример для ООП – **Фигура.** (Другой классический животные, гав-гав и мяу-мяу). Мы выбираем фигуру.
Фигура – корневой класс. Это **абстрактный** класс (Shape). Базовый класс.
Абстракция данных – это одно из **четырех** основных понятий ООП.
https://ru.wikipedia.org/wiki/Объектно-ориентированное_программирование

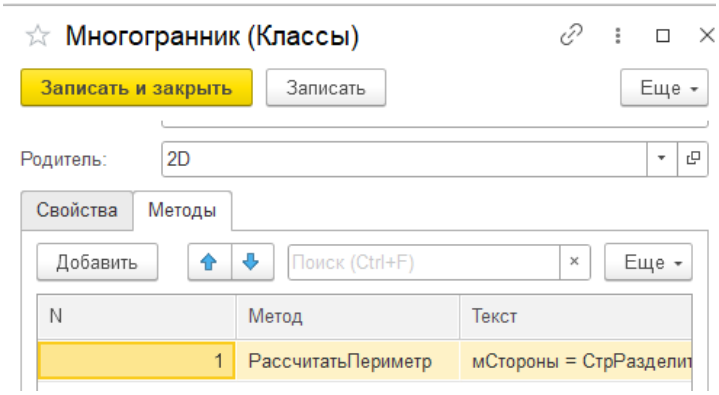
На основе класса **Фигура** – объекты не создаются. Он служит для передачи своих методов и свойств потомкам.



Класс **Многогранник**



Класс имеет одно свойство **Стороны** и один метод **РассчитатьПериметр.**



Метод:

```
мСтороны = СтрРазделить(Объект.Стороны.Значение,",");
ПериметрМногогранника = 0;
Для каждого Сторона Из мСтороны Цикл
    ПериметрМногогранника  = ПериметрМногогранника + Число(Сторона);
КонецЦикла;
Периметр = ПериметрМногогранника;
```

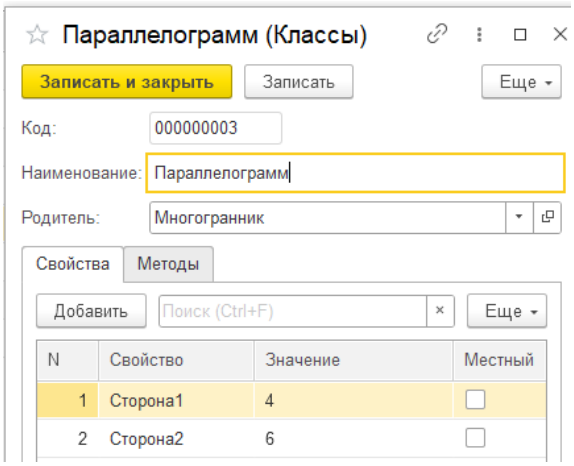
Для расчета периметра потомков можно было бы не добавлять (свойства и метод).
Почему не добавлять? Достаточно задать свойство **Стороны** следующим способом:

Параллелограмм	4,7,4,7
Прямоугольник	5,6,5,6
Квадрат	3,3,3,3
Треугольник	7,9,12

И использовать наследуемый метод **Многогранника**.

Но мы добавим и свойства, и методы, чтобы раскрыть возможности предлагаемой модели ООП в 1С.

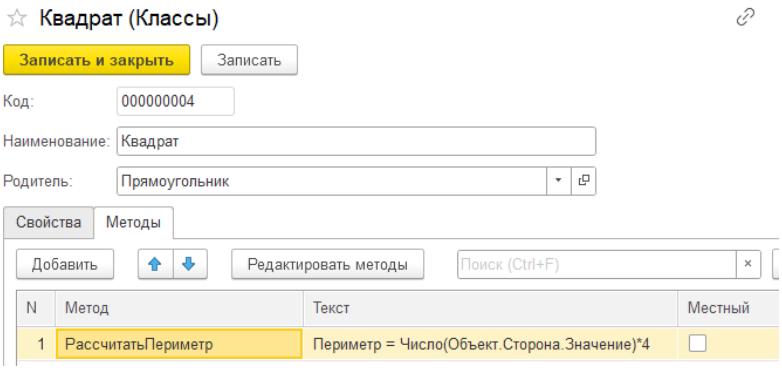
Класс **Параллелограмм**

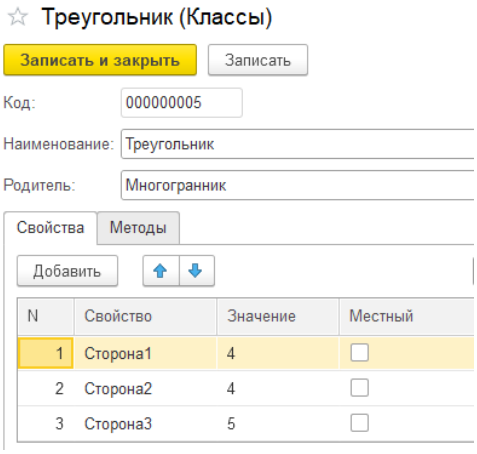


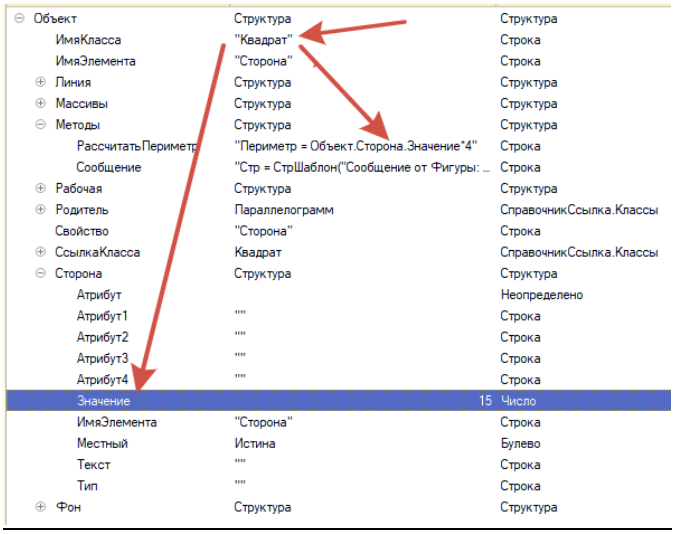
Метод:
Периметр = 2*(Число(Объект.Сторона1.Значение) + Число(Объект.Сторона2.Значение))

Метод наследуется классом **Прямоугольник**.

Для квадрата и треугольника свои свойства и методы.







Это картинка из расширенной версии ООП в 1С.
Видим в отладчике, что до **Квадрата** местные свойства **Прямоугольника** не дошли (инкапсуляция).
Есть свое свойство **Сторона** (значение по умолчанию = 15)
Метод «Сообщение» – достался от **Фигуры**, он наследуется всеми классами и используется для сообщений.
Есть свой метод расчета периметра.

К статье прилагается демонстрационная конфигурация.
В конфигурации рассмотрены примеры инкапсуляции, наследования и полиморфизма.

```
Функция ВычисленияНаСервере()
    Периметр = 0;

    //Ниже код без замашек на сходства с ОПП
    Объект = СоздатьОбъект("Многогранник");
    Выполнить(Объект.Методы.РассчитатьПериметр);
    ТекстСообщения = "Периметр многогранника " + Периметр;
    Выполнить(Объект.Методы.СообщитьПериметр);  //Метод унаследован от Фигуры


    //Ниже код с элементами синтаксиса ОПП
    Объект = СоздатьОбъект("Квадрат");
    Объект.Сторона.Значение = 15; //В классе задано значение по умолчанию = 7, меняем его
    Квадрат = Справочники.Классы; // уловка, чтобы методы вызывать через точку, как в ООП (оно и не надо в модуле менеджера процедура)
    Периметр = Квадрат.РассчитатьПериметр(Объект);
    ТекстСообщения = "Периметр квадрата " + Периметр;
    Квадрат.СообщитьПериметр(Объект,ТекстСообщения); //Указана в менеджере справочника Классы и использует метод унаследованный от Фигуры


    Объект = СоздатьОбъект("Прямоугольник");
    Объект.Сторона1.Значение = 15; //В классе вообще нет сторон, унаследовали. Поэтому по умолчанию нет, только назначать значения сторон
    Объект.Сторона2.Значение = 20;
    ОбъектПрямоугольник = Объект; // сохранили показать полиморфизм, так как переменная Объект переопределяется следующим текстом
    Прямоугольник = Справочники.Классы;
    Периметр = Прямоугольник.РассчитатьПериметр(Объект);
    Прямоугольник.СообщитьПериметр(Объект,"Периметр прямоугольника: " + Периметр);

    //Далее Полиморфизм. Здесь Многогранник вызвал метод Прямоугольника.
    //На практике для Полиморфизма лучше создавть отдельный класс. И к нему уже всем объектам. А здесь Дед к Внуку
    Объект = СоздатьОбъект("Многогранник");  //Объект, созданный выше переопределяется. ОбъектПрямоугольник будем использовать
    Многогранник = Справочники.Классы;
    ТекстСообщения = "Периметр многогранника: " + Многогранник.РассчитатьПериметр(Объект);
    Выполнить(ОбъектПрямоугольник.Методы["СообщаетПрямоугольник"]); //Полиморфизм - без уловки (см. выше)

    //Применим для расчета периметра метод Многогранника, когда стороны задаются в виде одной строки через запятую
    Объект = СоздатьОбъект("Многогранник");
    МетодРасчетаПериметраОтМногогранника = Объект.Методы.РассчитатьПериметр;
    Объект = СоздатьОбъект("Прямоугольник");
    Выполнить(МетодРасчетаПериметраОтМногогранника);
    //Здесь многогранник не передвл свойство детям (флаг Местный = Истина) - инкапсуляция
    //Но мы сами завели это свойство для прямоугольника

    ТекстСообщения = "Периметр прямоугольника, рассчитанный методом Многогранника " + Периметр;
    Выполнить(Объект.Методы["СообщаетПрямоугольник"]);

КонецФункции
```

Приведен пример расчета общего периметра группы фигур.

```
Процедура РасчетПериметровФигурНаСервере()
    //Значения сторон наследуются
    Выб = Справочники.Классы.ВыбратьИерархически(Справочники.Классы.НайтиПоНаименованию("2D"));
    Периметр = 0;
    ПериметрИтого = 0;
    Пока Выб.Следующий() Цикл
        Объект = СоздатьОбъект(Выб.Наименование);
        Выполнить(Объект.Методы.РассчитатьПериметр);
        ПериметрИтого = ПериметрИтого + Периметр;
    КонецЦикла;
    ТекстСообщения = "Общий периметр фигур " + ПериметрИтого;
    Выполнить(Объект.Методы.СообщитьПериметр);
КонецПроцедуры
```

Здесь примечательно то, что в цикле отрабатывают методы каждой фигуры. Код выглядит очень лаконично.

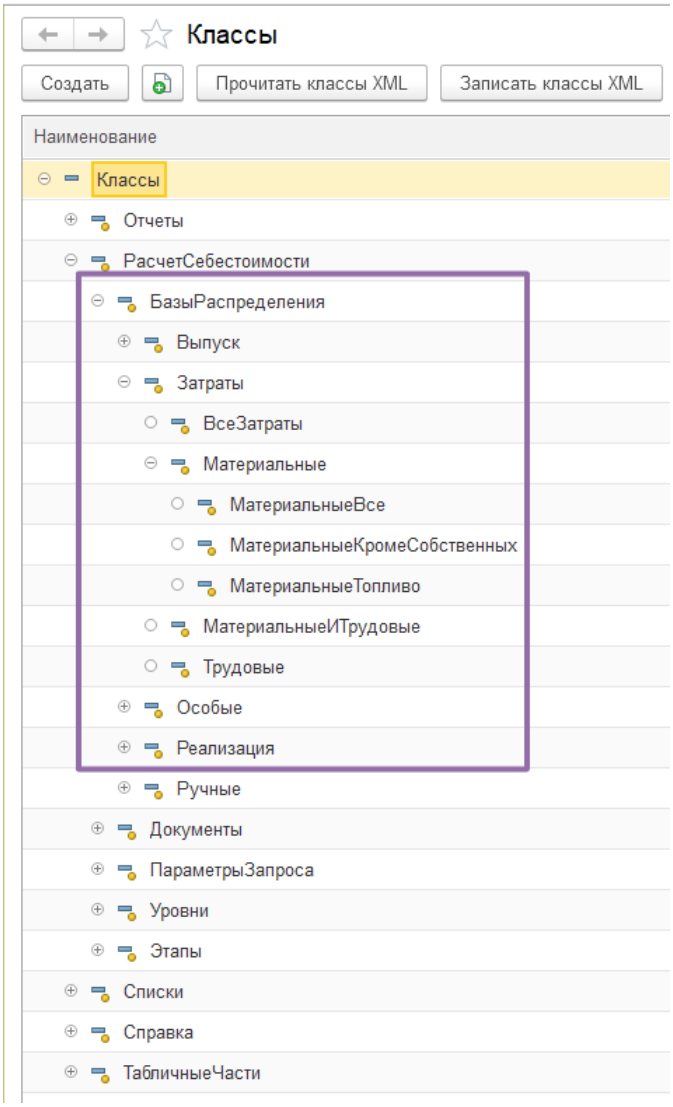
Конечно! Использование текста из справочника как алгоритма для команды **Выполнить** давно известно.
Но здесь не просто текст из справочника. Это метод **Объекта**. Наследованный или свой.

Структуры для ООП давно предложены, алгоритмы в текстовом поле справочника давно используются… Что же нового.
Немного юмора.
Попов. Радио. Патент. Ему говорят: резисторы, конденсаторы и индуктивности давно известны…

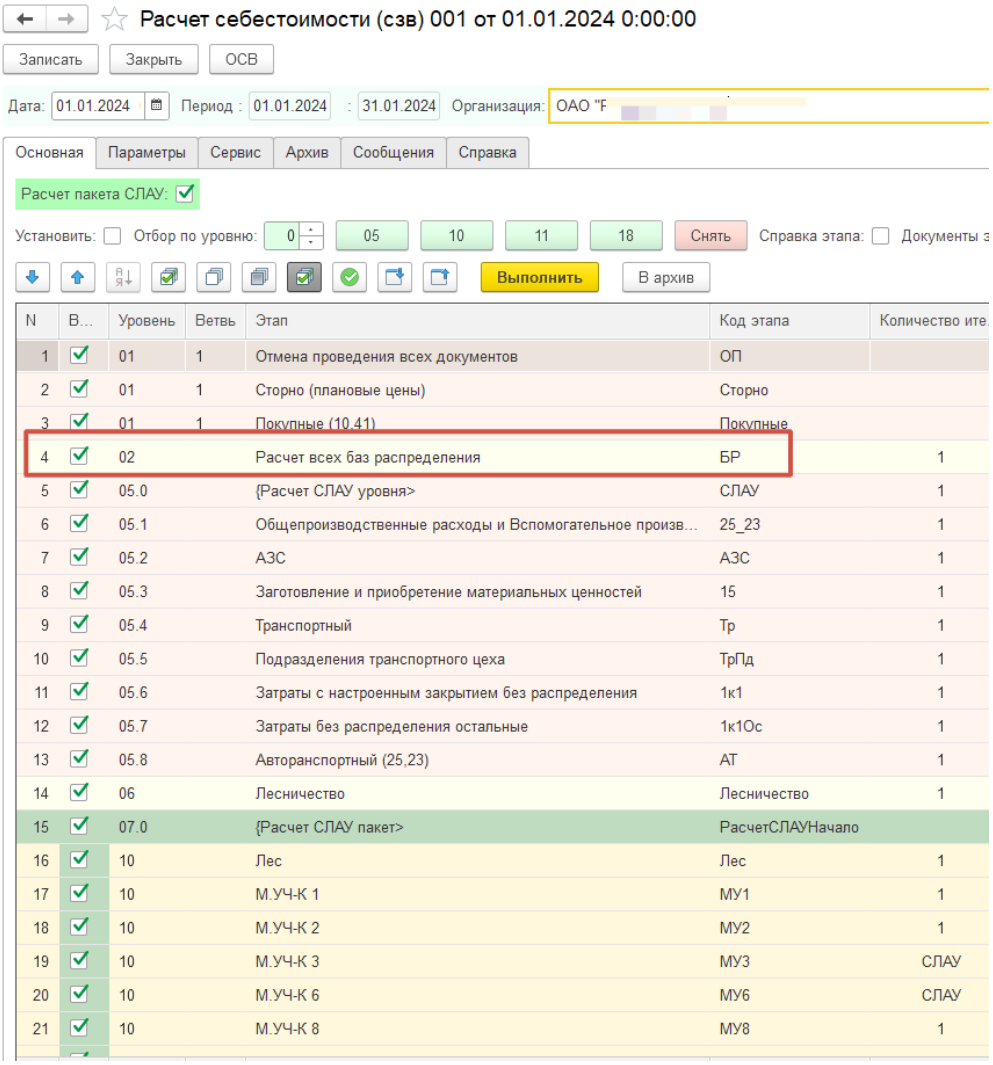
Nota bene – может не такая резиновая?

Как применить

Рассматриваемая в статье модель ООП применена в подсистеме расчета себестоимости. В настоящее время справочник **Классы** выглядит следующим образом:

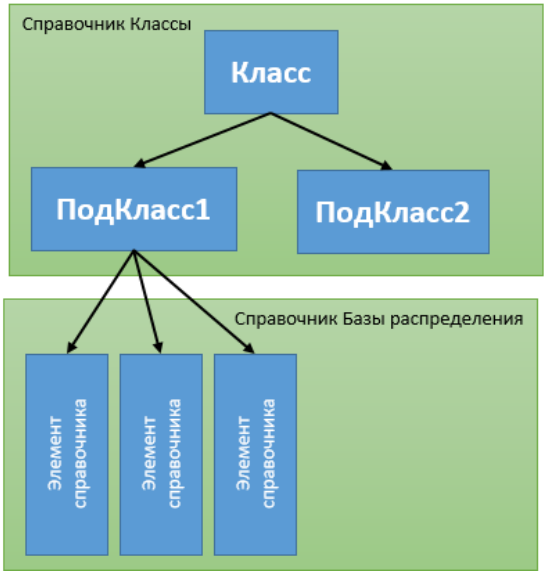


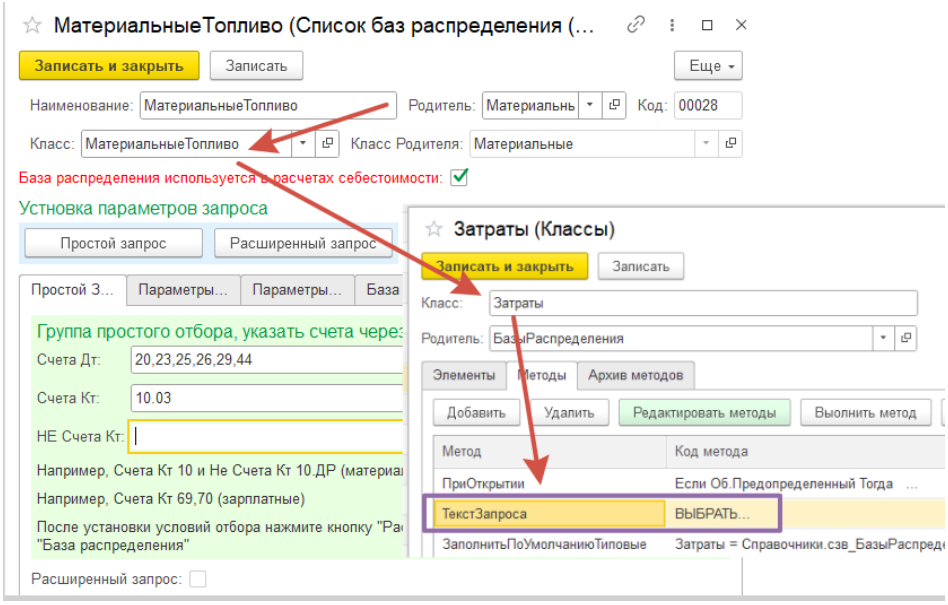
Этапы расчета себестоимости имеют вид:



N	В...	Уровень	Ветвь	Этап	Код этапа	Количество ите
1	☑	01	1	Отмена проведения всех документов	ОП	
2	☑	01	1	Сторно (плановые цены)	Сторно	
3	☑	01	1	Покупные (10,41)	Покупные	
4	☑	02		Расчет всех баз распределения	БР	1
5	☑	05.0		{Расчет СЛАУ уровня>	СЛАУ	1
6	☑	05.1		Общепроизводственные расходы и Вспомогательное произв...	25_23	1
7	☑	05.2		АЗС	АЗС	1
8	☑	05.3		Заготовление и приобретение материальных ценностей	15	1
9	☑	05.4		Транспортный	Тр	1
10	☑	05.5		Подразделения транспортного цеха	ТрПд	1
11	☑	05.6		Затраты с настроенным закрытием без распределения	1к1	1
12	☑	05.7		Затраты без распределения остальные	1к1Ос	1
13	☑	05.8		Авторанспортный (25,23)	АТ	1
14	☑	06		Лесничество	Лесничество	1
15	☑	07.0		{Расчет СЛАУ пакет>	РасчетСЛАУНачало	
16	☑	10		Лес	Лес	1
17	☑	10		М.УЧ-К 1	МУ1	1
18	☑	10		М.УЧ-К 2	МУ2	1
19	☑	10		М.УЧ-К 3	МУ3	СЛАУ
20	☑	10		М.УЧ-К 6	МУ6	СЛАУ
21	☑	10		М.УЧ-К 8	МУ8	1

Один из этапов – **Расчет всех баз распределения.**
На примере баз распределения рассмотрим применение ООП, хотя очень интересные эффекты обнаружились и для документов.
ООП в 1С даже интересней классического…
В классическом ООП каждый объект «живет» самостоятельно. Здесь же в 1С целесообразно использовать связь
Объекта 1С и объекта ООП. В элементе справочника (документе) есть ссылка на класс.
При работе с элементом справочника работаем с методами его (элемента) класса, которые унаследованы от предков.





Класс **МатериальныеТопливо** – наследует текст запроса от класса (дед) – **Затраты**.
В справочнике **Базы распределения** хранятся параметры для запроса. В этом случае Корсчет = 10.03.

В итоге процедура для этапа **Расчет всех баз распределения** имеет вид:

```
Функция РасчетВсехБР()
    Запрос.Текст = "ВЫБРАТЬ
    |   сзв_БазыРаспределения.Ссылка КАК Ссылка
    |ИЗ
    |   Справочник.сзв_БазыРаспределения КАК сзв_БазыРаспределения
    |ГДЕ сзв_БазыРаспределения.Используется";
    ТЗбр = Запрос.Выполнить().Выгрузить();
    Для каждого БР Из ТЗбр Цикл
        Объект = БР.Ссылка.ПолучитьОбъект();
        ОбъектБР = Классы.СоздатьОбъект(Объект.Класс.Наименование);
        Выполнить(ОбъектБР.Методы.РассчитатьБазуРаспределения);
    КонецЦикла;
КонецФункции
```

Лаконичный код.
Выбираются базы распределения, помеченные для использования.
Создаются два объекта (из двух справочников) и выполняется метод.

В итоге мы **Подсистеме расчета себестоимости** придали некую иерархичность. Разложили всё по «полочкам».
Модель легко масштабируется в ширину (братья и сестры) и в высоту (внуки и правнуки).
Но можно также (в глубину) между отцом и детьми добавить узел («отчим») и таким образом наследовать для детей всё наработанное для отца и его предков.